Feature extraction and classification of spam emails

Hassan, Muhammad Ali; Mtetwa, Nhamo

Published in:
2018 5th International Conference on Soft Computing & Machine Intelligence (ISCMI)

DOI:
10.1109/ISCMI.2018.8703222

Publication date:
2019

Document Version
Author accepted manuscript

Feature Extraction and Classification of Spam Emails

Muhammad Ali Hassan
Computer Science Department
COMSATS Universiry Islamabad, Lahore Campus
Lahore, Pakistan
mahassan@cuilahore.edu.pk

Nhamo Mtetwa
CCIS Department
Glasgow Caledonian University
Glasgow, Scotland
nhamoinesu.mtetwa@gcu.ac.uk

***Abstract*— Emails are a popular and preferred way of written communication in our daily life. The problem with emails is spam. These spam emails are sent with different intentions, but advertisement and fraud are the main reasons. As being inexpensive to send, it causes many problems to the internet society. This paper discusses the use of different feature extraction methods coupled with two different supervised machine learning classifiers evaluated using four performance metrics on two publicly available spam email datasets for spam filtering. We highlight the importance of the correct coupling of feature extraction and classifier, and the merits of using two independent datasets.**

Keywords: Spam Feature Extraction, Machine Learning, Support Vector Machine, Naïve Bayes

I. INTRODUCTION

The use of the Internet is on the rise. It has become an essential part of daily life. Emails are a powerful tool for exchanging information, ideas, and a preferred way of written communication. Emails can be sent to individuals or a group of people at the same time at no extra cost. Being cheap to send, secure and no time delays during transmission are some of emails' advantages. Despite these and many other advantages, there is an issue with spam emails. Spam is not only an annoyance, but it is also costly and can be a security risk which means it needs to be addressed.

According to the shortest and popular definition, spam can be defined as 'unsolicited' bulk email [1] usually commercial in nature. Spam emails are unwanted messages that are sent directly or indirectly, indiscriminately, having no contact or interaction with the user [2]. Spam emails are considered as spam because these emails are not valuable to the receiver. The main reason behind the growth of spam emails is advertisement of usually illegal, non-existent or worthless products, promotion of a cause, as bait for fraud schemes or delivery of malware. Only a small number of targeted users need to respond to these emails to make it profitable for the senders.

It is not obvious to tell who originally came up with the simple idea of sending out an advertisement to millions of email users such that at least one will respond to this advertisement no matter what is it all about. Because of this, the email accounts of millions of users are filled up with spam messages. Spam causes many problems to the Internet society; servers experience delays in the delivery of legitimate emails due to spam traffic. Within an organization, spam emails can affect: i) individual users, ii) reliability of emails, iii) productivity, iv) network bandwidth, v) computational power and storage space of servers [3]. According to specific respondents in a research report, filtering out spam emails can take up to 13 minutes on average of working time each day. [3]

The number of spam emails is generally increasing. Being inexpensive to send means, reaching millions of potential users at approximately no extra cost. According to a report, spam is mostly used to distribute phishing website links, malware and viruses [4]. The average number of spam emails sent is 54 billion throughout the world in each day [4]. Another statistical report by ACMA shows that 80,925 complaints have been received against spam emails in March 2017 [5].

II. LITERATURE

A. Corpus Data Pre-Processing

Before applying machine-learning algorithm, data pre-processing is necessary because not all information in an email is useful for spam classification. Removing specific noisy and less informative terms can enhance the performance of a classifier and decrease feature space dimensionality in most cases [6]. Data pre-processing is a set of steps used for transforming a specific dataset to a uniform style that is more conjusive for machine learning algorithms [7]. A corpus also called a dataset can have many features, so in order to avoid classifier over-fitting, one chooses only those features that enhance classification accuracy and help avoid over-fitting [8]. Feature Engineering (creating and defining only those features that make the classifier to perform better) is the basis of success and effectiveness of content-based spam filters. Outlined below are the steps used in the preparation of textual data for feature extraction:

Removal of special characters: Special character such as '#', '@' etc will be removed from the data during the pre-processing stage.

Tokenization: The textual data is broken down into individual words to identify specific words as spam or ham word. HTML tags, attachments, and headers are stripped rather than just emails subject and body line text. Domain name and IP addresses can also be considered as tokens.

Stemming: In this process, a different inflected form of words is grouped together e.g. plurals, gerund forms, tenses etc. For example, 'group', 'groups', and 'grouped' would all be considered as 'group'.

Case Conversion: Once the data has been tokenized, then words will be transformed to lower case.

Removal of stop words: Words like 'a', 'and' and 'the' etc are very common in English and are not helpful in identification of spam. These words are frequently used but they do not carry useful information for classification. Symbols and obscure text can also be removed in this step. Removal of such words can reduce the feature space and makes the classification more efficient.

Representation: This step involves the representation of textual data into a specific structured format that can be understood by the machine learning algorithm being used.

Lai and Tsai found that stop word removal results in better performance, but stemming does not have any significant impact on filters' performance. However, it results in feature size reduction [9].

B. Processing Pipeline and Feature Extraction

A feature in pattern recognition and machine learning can be defined as an individual characteristic or property of a phenomenon being observed [10]. It is the process of converting textual data into some specific format that can make the important text available for analysis. There are several feature extraction techniques for text classification e.g. bag-of-words, n-grams, TF-DIF etc.

1) Bag of Words

Bag-of-words is considered a popular and widely used approach for representing textual data in documents for analysis. It is easy and simple to understand and implement. In bag-of-words a textual document is represented as a bag-of-words simply means word frequency table or it means an unordered set of words keeping their frequency but ignoring their position in the document.

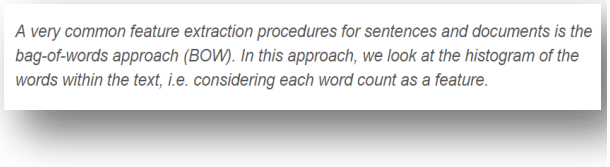

Figure 1: Bag-of-Words [11]

2) Term Frequency – Inverse Document Frequency (TF-IDF)

TF-IDF is a popular feature extraction approach used for document classification. A problem with the word count or term frequency is that words which are highly frequent start to dominate in the document, but such words may not contain a lot of information for the model as words which are rare but are domain-specific words. One technique is to re-scale the occurrence of words by how often these words occur in all the documents. Thus, the score for frequent words which may not contain much information across all documents are penalized. This technique of scoring documents is called TF-IDF in short, where:

Term Frequency (TF): The scoring of the frequency of a word in a current document.

Inverse Document Frequency (IDF): The scoring of the frequency of a word across all documents.

Suppose a document contains 100 words, where the word 'free' occurs 5 times. The term frequency can be can be calculated as:

$$TF(t) = \frac{Total\ no.of\ times\ term\ t\ comes\ in\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

$$TF(t) = \frac{5}{100} = 0.05$$

Now, assume we have 10,000 documents and the word 'free' appears 1000 times in all of them. Then, IDF is calculated as:

$$IDF(t) = \log\left(\frac{10{,}000}{1{,}000}\right) = 10$$

$$TF - IDF(t) = 0.05 * 10$$

$$TF - IDF(t) = 0.5$$

C. Machine Learning Algorithms

Machine Learning techniques have been successful for spam classification. These techniques extract information from labeled training datasets and use this information to train the classifier [12]. "The ability of computers to learn without being explicitly programmed" is called machine learning and it is somehow related to computational statistics and mathematical optimization [10]. Spam email classification is considered as a binary classification problem in which emails are classified as spam or ham. The machine learning algorithms that are most popular in spam classification are Naïve Bayes [13], Support Vector machine, Decision Tree and Neural network. These algorithms are more successful among other methods in the classification of spam emails [14].

1) Naïve Bayes

One of the popular classifiers used in spam classification problems is Naïve Bayes and it is also widely used in text categorization problems in general [7]. The Naïve Bayes classifier was initially proposed by [15] for spam classification problems. Naïve Bayes is based on the statistical Bayes Theorem. The main assumption in Naïve Bayes is that the features are independent.

The work by Sahami in 1998 [15] led to the implementation of several spam filters using different machine learning algorithms, in combination with the recommendations proposed by Graham in 2002 [16], employing Bayesian analysis. A comparison of Sahami and Graham model can be studied in a research report by Guzella & Caminhas [17].

Since the proposed work by Graham [16] and Sahami [15], several works have been proposed using the Naïve Bayes model. The main limitation of a simple Bayesian classifier is that it overlooks the association between words. For example, such a classifier doesn't account for the fact that words like 'special offers' more probably come together in spam emails than in ham emails. However, Orthogonal Sparse Bigrams (OSB) and Sparse Binary Polynomial Hashing (SBPH) were introduced to overcome this problem [18].

The SBPH technique was introduced by CRM114 and it is more expressive than OSB and takes a lot of memory and runtime overhead. SBPH is an inference of Bayesian filtering that can match individual words and mutating phrases. It uses Bayesian Chain Rule (BCR) to merge specifically individual feature conditional probabilities into a total probability. It has more expressive feature space and can deliver high accuracy. However, it is computationally expensive. On the other hand, OSB performs like SBPH but cuts most of the computational cost. OSB based filter together with other non-probabilistic algorithms as a replacement for BCR performed better than SBPH by 0.04% error rate [19].

Ben Medlock proposed Interpolated Language Model (ILM) in 2006 [20] that consider the structure of the email,

i.e. body and subject, in n-gram Bayesian model. The message classification depends on combining a static (Complete training dataset) and a dynamic (sometimes re-training with new dataset) element. The author compiled the GenSpam dataset himself for experimentation. The results achieved by ILM were better or at least like SVM using TF-IDF representation. It also surpassed Bayesian logistic regression and Multinomial Naïve Bayes [20].

Wang, Jones, and Pan [21] explored the performance of two online linear classifiers namely Winnow and Perceptron in 2006 with two datasets Ling-spam and PU1. They studied the performance with three feature selection methods, Document Frequency (DF), Information Gain (IG) and odds ratio. They observed that the performance of Perceptron and Winnow using IG and DF are better than using odds ratio. The result showed that Perceptron and Winnow both are very good classifiers for spam classification and their performance was better than Naïve Bayes [21].

Kim, Chung and Choi [22] used URLs in the email messages to filter spam messages using a Naive Bayes model. They used spam filtering based on URLs instead of analysing URL statistics to calculate the probabilities if the email with specific URLs is ham or spam. They concluded that the performance of their spam filter is similar, to those based on analysing keywords or URLs, but on the other hand, it does not need to maintain white or blacklists as in mostly URL based filters [22].

Segal [23] examined the performance of a Naïve Bayes classifier in 2007 that was trained either using messages from a specific group of users (personal messages) or messages from all users (global messages). The researcher found that the classifier trained using global messages performed better than the classifier trained from personal messages because of the larger training dataset. In this research, dynamic personalization was proposed and explored, and how it can help in the improvement of overall performance [23].

2) Support Vector Machine (SVM)

Support Vector Machine was used by Drucker, Wu, and Vapnik [8]. They used the bag-of-words representation with TF-IDF and binary representation, with two private datasets. The best results came from SVM using binary representation. SVM can be trained using fewer training examples and it can deal with multi-dimensional data using kernels [24]. It maps the training data to a feature space using kernel functions. This feature space then separates the samples with maximum margins by generating hyper planes which can be used as a non-linear decision boundaries. Rios and Zha [24] used private and public emails from the Internet collected over several years, with two state-of-the-art algorithms including Support Vector Machines for spam classification. They stressed the importance of feature engineering and their results showed that SVM significantly performed better than Naïve Bayes [24].

Bickel and Scheffer [25] designed a framework using publicly available labelled and unlabeled datasets for training a classifier. According to the experimental results achieved with binary representing spam messages from different sources and Enron corpus, it was seen that their proposed methodology decreased risk by about 40% as compared to the use of single classifier [25].

Kanaris [26] trained a linear SVM using n-gram characters, employing the IG for feature selection. The datasets used in this research for experimentation were Ling-spam and SpamAssassin [26] and were based on cross-validation and n-gram models. The results achieved while experimenting with an n-gram based model was better or comparable to the word-based representation [26].

Amayri and Bouguila [27] investigated several-distance based kernels and behaviors of spam filters using SVM. They observed that most of the kernels used in recent researches overlook the structure of the text. Instead of using these classical kernels, they proposed string-based kernel for spam classification. They showed how efficient these string kernels are for spam classification problems. Their results from an extensive study showed that active online methods using string kernels achieved high recall and precision scores [27].

D. Performance Metrics

With so many supervised machine learning classifiers available there is need for a way to evaluate their classification capability. In this paper we use four performance metrics: accuracy, precision, recall and f1-score. Accuracy is the most intuitive performance measure: it is simply a ratio of correctly predicted observations to the total observations. Accuracy is a great measure but only when one has symmetric datasets. Therefore, one has to look at other metrics to evaluate the performance of their model. Our results consist of precision, recall and F1 score. Precision is the ability of a classifier not to label a positive sample as negative or label a negative sample as positive. Recall is the ability of a classifier to find all the positive samples in a dataset. F1 score is the harmonic mean of precision and recall.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

True Positives (TP): Such cases in which we predicted yes (email is spam), and it was spam. **True Negatives (TN):** Such cases in which we predicted no (email is not spam), and email was not spam. **False Positives (FP):** We predicted yes (email is spam), but in the actual email was not spam. (Also known as a "Type I error."). **False Negatives (FN):** We predicted no (email is not spam), but in the actual email was spam. (Also known as a "Type II error.")

III. METHODOLOGY

The following figure shows the methodology that has been used in this work.

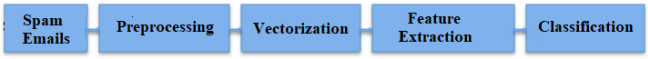

Figure 2: Flowchart of methodology

A. Dataset

We combined two different datasets for experimentation. The first dataset is extracted from Ling-spam corpus, which is a popular email dataset and has been widely used by many researchers [28]. The Ling-spam dataset can be found here [29]. The second dataset s extracted from Enron, it is also a popular dataset that has been widely used by many researchers. The raw Enron dataset is available at [30].

The dataset used in this research is a combination of Ling-spam and Enron Corpus. The reason for not only using Ling-spam corpus is that there were not enough spam emails present in that dataset, but the number of ham emails were quite higher as compared to the number of spam emails and using such dataset where the class distribution of one class is higher will result in biased classifier towards one class. So, some of the spam and ham emails are taken from Ling-spam corpus and remaining ham emails are taken from Enron corpus. The total number of emails in our dataset is 2457, out of which 1312 are ham and 1145 are spam emails. Figure 3 shows the distribution of the spam and ham emails. It can be see that the dataset is fairly balanced between ham and spam emails.

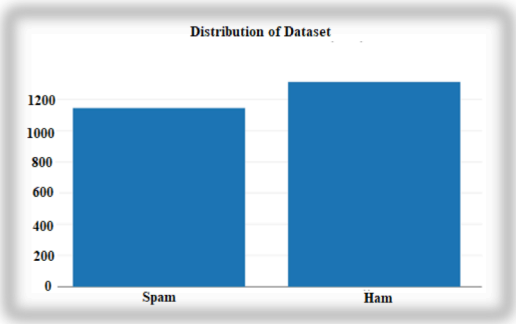

Figure 3: Distribution of Dataset

B. Processing Pipeline and Feature Extraction

The raw data is loaded and pre-processed by removal of special characters, tokenization, stemming and removal of stop words. A Python regular expression library is used for the removal of special characters while Natural Language Toolkit (NLTK) is used for tokenization, stemming and removal of stop words. Pre-processing achieves text normalization. Text normalization is the process of transforming text into a specific format appropriate for feature extraction. After normalizing the textual emails data, we create features using bag-of-words and TF-IDF vectorization techniques. After suitably transforming the features for each of the two machine learning algorithms we train the models using the training dataset and then test using the testing dataset. The dataset was split 70% for training, 20% for testing and 10% for validation. Two different experiments are performed using the dataset mentioned above and their results are compared. We evaluate each trained model using four performance metrics namely accuracy, recall, precision and F1-score.

IV. Results

The table in figure 4 shows the results of two supervsed machine learning algorithms SVM and Naïve Bayes along with two different feature extraction techniques bag-of-words and TF-IDF applied to these machine learning classifiers being scored against each other using four performance metrics.

Features			**Bag-of-Words**	**TF-IDF**
SVM	Accuracy		0.93	0.99
	Precision	Ham	0.94	0.98
		Spam	0.73	1.0
	Recall	Ham	0.98	1.0
		Spam	0.48	0.98
	F1	Ham	0.96	0.99
		Spam	0.58	0.99
Naïve Bayes	Accuracy		0.80	0.98
	Precision	Ham	0.96	0.98
		Spam	0.30	0.98
	Recall	Ham	0.82	0.98
		Spam	0.71	0.98
	F1	Ham	0.88	0.98
		Spam	0.42	0.98

Figure 4: SVM and Naïve Bayes using BoW and TF-IDF

As it can be seen from the above table, Linear SVM performed better, it achieved 93% accuracy, whereas Naïve Bayes achieved 80% accuracy on our dataset using bag-of-words approach. Looking only on the accuracy cannot be very helpful to evaluate the performance of the classifier, whereas confusion matrix gives us more deep insights into the performance of the classifiers. As it can be observed that both the classifiers e.g. SVM and Naïve Bayes managed to produce higher Precision and Recall values for ham emails as compared to spam, which means that the classifier manages to predict most of the ham emails correctly, but they did not predict many of the spam emails as spam using bag-of-words approach. Recall is high, which means that the classifier managed to predict most of the ham emails correctly, but it wasn't able to predict many of the spam emails as spam.

On the other side, both the algorithms SVM and Naïve Bayes achieved very high accuracies using TF-IDF, which is a good thing but at the same time suspicious as well, because the SVM achieved over 99% accuracy and around 98% accuracy was achieved with Naïve Bayes using TF-IDF as feature selection method, that can be a sign of an overfitted model.

V. Conclusion

This paper discussed the use of different feature extraction methods coupled with two different supervised machine learning classifiers evaluated using four performance metrics on two publicly available spam email datasets for spam filtering. Previous research has considered these datasets separately. We highlighted the importance of the correct coupling of feature extraction and classifier, and the merits of using two independent datasets. In future work these results could be further analysed using cross tabulation techniques and also investigate possible overfitting.

REFERENCES

[1] I. Androutsopoulos, J. Koutsias, K. Chandrinos and C. D. Spyropoulos, "An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages," *Computation and Language,* pp. 160-167, 2000.

[2] G. V. Cormack, "Email Spam Filtering: A Systematic Review," *Foundations and Trends® in Information Retrieval,* vol. 1, no. 4, pp. 335-455, 2006.

[3] M. Siponen and C. Stucke, "Effective Anti-Spam Strategies in Companies: An International Study," *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06),* 2006.

[4] Cyberoam, "Internet Threats Trend Report," 2014. [Online]. Available: https://www.cyberoam.com/downloads/ThreatReports/CyberoamCYRENInternetThreats2014April.pdf.

[5] A. C. a. M. Authority, "Spam Statistics," 2017. [Online]. Available: https://www.acma.gov.au/theACMA/ACMAi/Investigation-reports/Statistics/spam-statistics.

[6] Y. Diao, H. Lu and D. Wu, "A Comparative Study of Classification Based Personal E-mail Filtering," *Lecture Notes in Computer Science,* vol. 1805, pp. 408-419, 2000.

[7] L. Zhang, J. Zhu and T. Yao, "An evaluation of statistical spam filtering techniques," *ACM Transactions on Asian Language Information Processing (TALIP) TALIP,* vol. 3, no. 4, pp. 243-269, 2004.

[8] H. Drucker, D. Wu and V. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural Networks,* vol. 10, no. 5, pp. 1048-1054, 1999.

[9] C.-C. Lai and M.-C. Tsai, "An empirical performance comparison of machine learning methods for spam e-mail categorization," 2004.

[10] Wikipedia, "Machine Learning," 2017. [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning.

[11] Y. Goldberg, "Bag-of-Words approach," in *Neural Network Methods in Natural Language Processing (Synthesis Lectures on Human Language Technologies)*, G. Hirst, Ed., Morgan & Claypool Publishers, 2017, p. 310.

[12] P. Louridas and C. Ebert, "Machine Learning," *IEEE Software,* vol. 33, no. 5, pp. 110-115, 2016.

[13] T. Sun, "Spam Filtering based on Naive Bayes Classification," 2009.

[14] T. A. Almeida and A. Yamakami, "Content-based spam filtering," *The 2010 International Joint Conference on Neural Networks (IJCNN),* 2010.

[15] M. Sahami, S. T. Dumais, D. Heckerman and E. Horvitz, "A Bayesian Approach to Filtering Junk E-Mail," 1998. [Online]. Available: http://aaai.org/papers/workshops/1998/ws-98-05/ws98-05-009.pdf. [Accessed 20 11 2017].

[16] P. Graham, "A Plan for Spam," 2002. [Online]. Available: http://www.paulgraham.com/spam.html. [Accessed 20 11 2017].

[17] T. S. Guzella and W. M. Caminhas, "Review: A review of machine learning approaches to Spam filtering," *Expert Systems With Applications,* vol. 36, no. 7, pp. 10206-10222, 2009.

[18] C. Siefkes, F. Assis, S. Chhabra and W. S. Yerazunis, "Combining winnow and orthogonal sparse bigrams for incremental spam filtering," *Lecture Notes in Computer Science,* pp. 410-421, 2004.

[19] A. Bhowmick and S. M. Hazarika, "Machine Learning for E-mail Spam Filtering: Review, Techniques and Trends," 2016.

[20] B. Medlock, "An Adaptive, Semi-Structured Language Model Approach to Spam Filtering on a New Corpus.," *The Third Conference on Email and Anti-Spam, July 27-28, 2006, Mountain View, California, USA,* 2006.

[21] B. Wang, G. J. F. Jones and W. Pan, "Using online linear classifiers to filter spam emails," *Pattern Analysis & Applications,* vol. 9, no. 4, pp. 339 - 351 , 2006.

[22] J. Kim, K. Chung and K. Choi, "Spam Filtering With Dynamically Updated URL Statistics," *IEEE Security & Privacy,* vol. 5, no. 4, 2007.

[23] R. Segal, "Combining Global and Personal Anti-Spam Filtering," 2007.

[24] G. P. Rios and H. Zha, "Exploring Support Vector Machines and Random Forests for Spam Detection.," 2004.

[25] S. Bickel and T. Scheffer, "Dirichlet-Enhanced Spam Filtering based on Biased Samples," 2007.

[26] I. Kanaris, K. Kanaris, I. Houvardas and E. Stamatatos, "Words versus Character n-Grams for Anti-Spam Filtering," 2007.

[27] O. Amayri and N. Bouguila, "A study of spam filtering using support vector machines," *Artificial Intelligence Review,* vol. Volume 34, no. 1, p. 73–108, 2010.

[28] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras and C. D. Spyropoulos, "An Evaluation of Naive Bayesian Anti-Spam Filtering," *Computation and Language,* 2000.

[29] C. GROUP, "Ling-Spam Dataset," 2010. [Online]. Available: http://csmining.org/index.php/ling-spam-datasets.html..

[30] C. GROUP, "Enron-Spam dataset," 2010. [Online]. Available: http://csmining.org/index.php/enron-spam-datasets.html.